Kelsey Heidarian
CS 5959 Writing Solid Code
Coreutils Evaluation – numfmt

`numfmt` reads numbers in various representations and reformats them as requested. The most common usage is converting numbers to/from *human* representation (e.g. ‘`4G`’ → ‘`4,000,000,000`’).
`numfmt [`*`option`*`]… [`*`number`*`]`

`numfmt` converts each *number* on the command-line according to the specified options (see below). If no *number*s are given, it reads numbers from standard input. `numfmt` can optionally extract numbers from specific columns, maintaining proper line padding and alignment.

An exit status of zero indicates success, and a nonzero value indicates failure.

Testing: I performed differential testing for numfmt. In order to avoid hammering improper formatting checks, I randomly generated large numbers of varying length and appended a char or two at the end of the number so that it would match the input specification more less. I wanted to make sure that my test was able to access deeper in the code. I ran the same command using an older version and compared the output from that to the output from the systemwide numfmt. If a difference was found I logged the offending command along with the random number generator seed and all other pertinent data. Unfortunately, I was unable to find any cases where the output was different.

Although, I attempted to restrict my random input to better access more of the source code I ended up achieving only 53.05% code coverage. numfmt.c is quite long at 1535 lines of code. numfmt crashes fairly easily when the input is too long. It exits the program with an OSError that reads: “Argument list too long”.